Large Language Models for Small Data of Industrial Domains

Bhavna Agrawal[1], Anuradha Bhamidipaty[2], Roy Abitbol[3], Eyal Cohen[4]
[1,2]IBM Research, Yorktown Heights, NY, [3,4]IBM Research, Haifa, Israel,
{[1]bhavna,[2]anubham}@us.ibm.com, {[3]Roy.Abitbol,[4]eyalco}@il.ibm.com

Abstract

Classification of industrial data sets remains a challenging problem because of the high domain specificity of the data and the lack of enough labeled data. Many approaches, including many zero-shot techniques, perform poorly due to the domain specificity of the data. In this work, we present an *novel automated pipeline* that is configurable and extensible to address the industrial environments concerns like domain specificity and lack of labeled data. The pipeline consists of 1) synthetic data generation (SDG) techniques that retain context sensitivity, 2) evaluation of synthetic data, and 3) fine-tuning an encoder-only transformer-based classifier. Our results demonstrate over 40% improvement in the accuracy of the classification with small data set (less than 10 samples per class) when using domain-specific datasets as well as generic public datasets.

A template-based, Kmodels, approach is used for model deployment, which allows easy configuration of the pipeline, enabling custom multi-site deployment. This approach was validated through the deployment at an actual client site, and resulted in improved efficiency.

1 Introduction

Within an asset management environment, work orders are created to resolve incidents as they occur. However, the technician must choose a problem code from many choices, and for expediency, they end up choosing "miscellaneous". In general, the choice of this problem code does not hinder the work of the technician but impacts the data quality, and thus the operations decision making. For the same reason, the amount of labeled data (ground truth) is limited in real industrial use cases. In addition, these industrial work orders contain domain-specific jargon that renders zero-shot techniques ineffective. A typical sample for Oil & gas industry might look like:

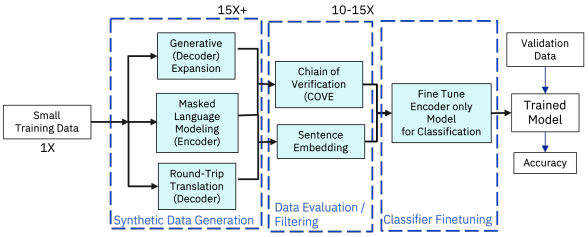

Figure 1: Pipeline for multi-class classification of small industrial data.

```
Replace damaged finger in the lower
fingerboard, not closing on demand.
```

In this work, we present a novel automated pipeline, Figure 1, for the classification problem. The pipeline consists of 1) synthetic data generation techniques that retain the context sensitivity, 2) evaluation of synthetic data, and 3) fine-tuning an encoder-only transformer-based classifier. For a typical industrial domain, the input data may contain only 5-10 samples per label/class and may contain a lot of industry and organization-specific jargon. Therefore, synthetic data generation techniques need to be selected very carefully to ensure high contextual fidelity.

Once a pipeline is defined, it is encapsulated within a model *template* along with all its parameters and configuration information, overlayed with KubeFlow Pipeline and KServe for training, and inferencing the models respectively. This encapsulation is a key concept of KModels deployment methodology (Abitbol et al., 2024). As illustrated in figure 2, a single template can be uniquely instantiated using varied configurations for different datasets, and different compute environments.

To that effect, we shall describe in detail the two main aspects of this work: 1) Pipeline for the multi-class classification of industrial data, 2) Deployment in an industrial environment, followed by the results and conclusions. But first we describe the related work.

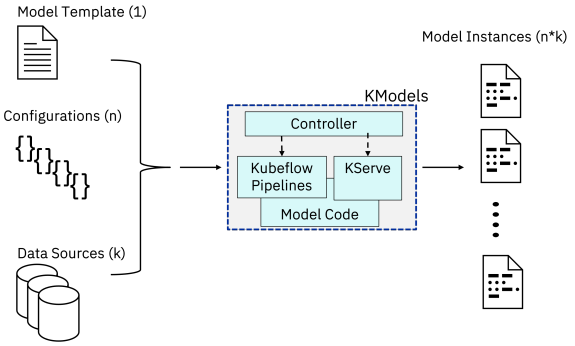

Figure 2: High-level view of KModels, the framework that allows for simple, scalable deployment of models at multiple customers sites.

2 Related work

Domain-specific contextualization of the BERT (Devlin et al., 2018) like language model by continued pretraining of the model has been very successful in some fields, *e.g.* medical (Rasmy et al., 2021) and legal (Chalkidis et al., 2020) because 1) there are large corpora of data available for each of these domains, and 2) there is a standard vocabulary across the domain. However, for an industrial domain such is not the case because neither the data is available in such large quantities, nor the vocabulary is standardized across organizations or industries.

A good survey of data augmentation methods for NLP published before 2021 can be found in (Feng et al., 2021). We shall also lean on some of the methods outlined there in our ensemble of data augmentation techniques, like back translation (Sennrich et al., 2015), also commonly known as Round Trip Translation, or RTT, and context enabled data augmentation (Kobayashi, 2018). Some of the other techniques, like adding noise to the data (Wei and Zou, 2019), or substituting dictionary-based synonyms (Wei and Zou, 2019) etc. are not as useful because our downstream application, multiclass classification, requires context-sensitive augmentation of the data.

Recently, there has been an interest in creating classifiers by first generating a labeled data set given the classification categories and then fine-tuning the classifiers. These are called zero-shot classifiers, for example (Meng et al., 2022). In (Ye et al., 2022), they trained classifier on given standard data set to verify the accuracy of the generated data set. However, in both of these cases, classification categories are from generic topics with abundant public-domain data, which was likely seen by generative LLM during pretraining, with no opportunity to do domain- or organization-specific data generation.

In fact, for domain-specific data (IT Support ticket domain), Wahba *et al.* (Wahba et al., 2023) show that the generic embeddings are not very good at representing their data, and thus one can use a simpler SVM model for classification with equivalent or better performance.

Another approach to creating a high-quality classifier from a limited amount of labeled data is to do consistency training (Xie et al., 2020; Chen et al., 2020) on the augmented unlabeled data where augmentations are performed using a very reliable method like round-trip translation. This again assumes availability of significantly large amount of unlabeled data for each category which may not be the case for our application (*e.g.* see `Dataset 3` defined later).

There are not many mentions of data filtering approaches in literature because most of the papers evaluate the quality of the data augmentation using the performance of the ultimate natural language task (multi class classification in our case). However, a popular approach to contain the hallucinations of the LLM has been to use "Chain of Thought" (Wu et al., 2023) which asks the LLM's to think "step by step". Based on this idea, another approach was suggested in (Dhuliawala et al., 2023), called "Chain of Verification" (COVE), which essentially uses LLM's to verify its own answers. If an inconsistency is observed, it has an opportunity to correct itself. We utilize a version of this approach as one of the methods to verify the correctness of our data augmentation using LLM's.

To the best of our knowledge, the work described in this paper is the first implementation of a configurable and extensible pipeline that meets both the domain specificity and variability of the industrial data sets, and has also been demonstrated on public data set. In addition, newer synthetic data generation methods, evaluations methods, and classifiers may be added to the framework as needed.

3 Datasets

We have carefully chosen data sets to illustrate the typical qualities of data in industrial domain.

1. `Dataset1`: A public dataset[1] with 20 classes (news groups), uniformly distributed across all

[1] `[1]http://qwone.com/~jason/20Newsgroups/`

classes, 1000 samples/class; Used for benchmarking.

2. `Dataset2`: Real estate industry data, with 38K samples across 53 categories, in 6 different languages; the data is mostly reviewed and labeled data; it contains some, though not overwhelming, amount of labeling noise. It is an example of *skewed data set*, with thousands of samples for some classes, and 1-5 samples for others.

3. `Dataset3`: Oil & Gas Industry data with 223 samples across 15 classes; reviewed and labeled data. It is an example of *not enough data* for almost any of the categories, the data ranges from 45 samples to 5 samples per class. Also includes *industry-specific jargon*, *e.g.* `"Replace Relief Valve on Acoustic detector on SRS#1."` does not mean much to an average person, but in oil and gas industry, it means "replace the relief valve on the acoustic detector on safety release system (SRS)" and it is an example of "Parameter Deviation" problem code.

4 Synthetic Data Generation (SDG)

We apply an ensemble of data generation approaches to achieve a large amount of synthetic data that can be filtered to generate high-quality data. The three approaches we use are:

1. Generative LLM Based Data Augmentation
2. Round Trip Translation (RTT)
3. Masked Language Modeling (MLM)

4.1 Generative LLM Based Data Augmentation

Data is generated for each class separately based on the ground truth examples and the label description. We typically use LLM to augment each sample with 10 additional samples. We experimented with two different kinds of prompts: 1) Simple prompt 2) Detailed instructions. In addition, we added domain information and the definition of the problem code for each of these types of prompt to further help data generation.

The simple prompt is:

```
Please rephrase the following sentence. DO NOT provide a solution in your response.
```

The "Detailed Instruction" prompt is:

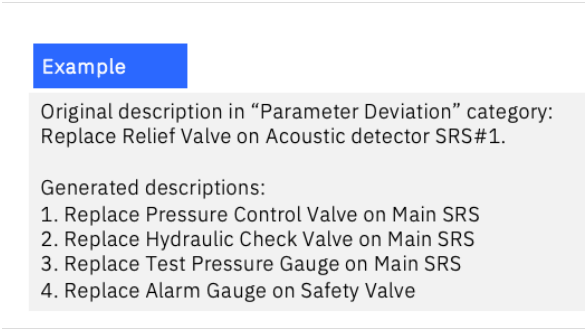
Example
Original description in “Parameter Deviation” category:
Replace Relief Valve on Acoustic detector SRS#1.
Generated descriptions:
1. Replace Pressure Control Valve on Main SRS
2. Replace Hydraulic Check Valve on Main SRS
3. Replace Test Pressure Gauge on Main SRS
4. Replace Alarm Gauge on Safety Valve

Figure 3: Context sensitive data augmentation

```
The following sentence presents a sample problem record in the operations. Please generate a new record following the same pattern of the sample record. One way to generate another pattern is to change the equipment name. DO NOT provide a solution in your response.
```

One may include the domain in the prompt by changing "operations" to "Oil & Gas industry operations" or "real estate industry operations".

Finally, a one-shot example of the class we are generating data for may be included with either of the above prompts:

```
Problem code: Parameter Deviation
Sample record: Replace Relief Valve on Acoustic detector on SRS#1.
```

An example of expansion from detailed instructions with problem code definition is shown in Figure 3, clearly the augmentation maintains the context very faithfully.

4.2 Round Trip Translation

Back translation techniques were first introduced by Sennrich et. et (Sennrich et al., 2015) for creating additional training data for machine translators and improving their performance, especially if the data was sparse for one of the languages. Round-trip-translation (RTT) has long been used as a data augmentation technique in general. After experimenting with a number of languages, We settled on 4-different languages based on their performance: French, Spanish, German, and Portuguese. In fact, back translation technique has in general been used where one needs to reproduce high-fidelity data compared to the original one. This is the reason why we chose this as one of the techniques for our context-sensitive synthetic data generation. We use the LLM to translate sentences back and forth.

Table 1 shows the results for `Dataset2` for Round trip translation. As explained in Section 5, it maintains the context fidelity very well (91% accepted expansions), and shows improved classification accuracy. However, we run the risk of over-fitting if we use only this method of data augmentation.

Augmentation Method	Accepted Expansions	Accuracy
Baseline	-	0.18
RTT	91%	0.55
MLM	99%	0.60

Table 1: Results of RTT and MLM methods of data augmentation.

4.3 Masked Language Modeling (MLM)

Kobayashi et. al (Kobayashi, 2018) first introduced the concept of switching contextually similar words for data augmentation (as opposed to dictionary based synonyms), and achieved good results. We use a very similar approach, where we mask words in the original description and ask LLM to generate the masked words. Multiple words are masked per sample, but only one word is masked at a time, and multiple replacements are sought for each masked word. We use Roberta (Liu et al., 2019) based models for mask prediction. The results are shown in Table 1 for MLM. As expected, it produces very good quality synthetic data (99% acceptance rate).

5 Data Evaluation / Filtering

Data Filtering of the synthetically generated data has proven to be a key to maintaining the quality of training data and thus the quality of the classifier. We use two approaches to evaluate the results of the data augmentation:

5.1 Sentence embedding comparisons

We use the cosine similarity of the sentence embeddings from sentence transformers of the two samples (original description and augmented description) to compute the contextual similarity of two phrases. The acceptable similarity threshold was chosen empirically such that the phrases were not identical, but not too far apart either. While it works well in most cases it could not identify the samples from Figure 3 as being acceptable augmentations.

5.2 Chain of Verification

The other approach to verification is using the COVE approach described in (Dhuliawala et al., 2023). The idea of this approach is to use an LLM to verify the correctnness of the generated response. Here, we ask the LLM if the augmented response is a valid response. We tried many prompts, but this one was the best:

```
Given the problem code, the sample record,
and test record, identify if the test record
represents an additional sample of the sample
record. Please respond with a 'yes' or 'no'.
Problem code: Parameter Deviation
Sample record: Replace Relief Valve on
Acoustic detector on SRS#1.
Test Record: Replace Pressure Control
Valve on Main SRS
```

These two methods may be combined together in a number of ways, such as majority voting, weighted average of the two votes, or staggered vote, depending on the knowledge of the data. This could be a configuration parameter of the pipeline as described in Section 8. The acceptance rates of individual filtering mechanisms for the augmented data, where COVE uses detailed prompt with domain specification and class definition, are shown in Table 2. As we can see that the acceptance rates are similar for both methods.

data	sentence similarity	COVE
Detailed instr. (dataset 2)	87%	83%
Detailed instr. (dataset 3)	84%	80%

Table 2: Comparison of two filtering mechanisms for augmented data.

6 Classification

We fine tuned an encoder only transformer-based Roberta model (Liu et al., 2019). We did an internal evaluation of the fine-tuning of multiple classifiers and concluded that the accuracy is not very different across different fine-tuned models for our use case. Details of classifier model fine-tuning are: learning rate of 2e-5, batch size of 8, and epsilon of 1e-8 for Adam optimizer. We use 10 epochs for fine-tuning. The models are trained using 32 cores and 2GPU.

7 Pipeline

We tested our pipeline for three different LLM's as shown in Table 3. The classifier accuracy was computed on partial data from `Dataset 2`. We saw similar results for the other data sets as well. Interestingly, Google/flan-ul2 outperforms many of the newer and larger models. This is partly explained by the fact that we want generation to be quite close to the original samples to maintain contextually fidelity. Therefore, going forward, we shall use Google/flan-ul2 for all our experiments.

LLM	Classifier Accuracy
Meta/Meta-Llama-3-70B-Instruct	0.26
Mistralai/Mixtral-8x7B-Instruct-v0.1	0.21
Google/flan-ul2	0.52

Table 3: Comparison of different LLMs for our pipeline.

Next we present the results of the overall pipeline for a number of datasets. First we show the results on the public dataset in Table 4. The classes are newsgroups that have discussions on generic topics such as religion, computer hardware, space, etc. This does not compare directly with our datasets because of the lack of domain specific jargon, but this is the closest public data set we have to our work. We divided the data into train and test (80/20), and within the train data, we further reduced the sample size artificially, to 1%, 2%, etc. and ran our pipeline with and without Synthetic Data Generation (SDG) pipeline, using all three techniques. As we can see, just 8 samples/class with SDG give an accuracy similar to 50-60 samples/class without SDG

Table 5 shows the results of only the *generative LLM* of SDG on two data sets, `Dataset 2` (Real Estate) , and `Dataset 3` (Oil & Gas). `Dataset 3` is a very small sample set (115 train samples across 15 classes), and contain a lot of industry-specific jargon. Thus, the macro-f_1 average is only 0.18 on the original data, and increases to 0.61 with the SDG pipeline. Additionally, the expansion works best when one includes the problem code definition and domain in the prompt. We discuss the results of `Dataset 2` next.

Table 6 shows the results of two classifier fine-tuned for `Dataset2` and `Dataset3` with all three methods of data augmentation. `Dataset 2` does not show any improvement in the accuracy because

% of train data	Original samples/ class	No Synthetic Data	**With** Synthetic Data
1	8	0.15	0.66
2	16	0.39	0.69
3	24	0.54	0.74
5	40	0.57	0.75
10	80	0.74	0.81
50	400	0.87	...[a]
100	800	0.90	...[a]

[a] ... requires no synthetic data generation.

Table 4: Classification accuracy of our pipeline on a public dataset. Just 8 samples/class with SDG give an accuracy similar to 50-60 samples/class without SDG.

the results are dominated by the larger classes, and only 180 samples out of 34K samples were used for SDG (classes with <100 samples). However, for 7 of the smallest classes of this data, macro-f_1 average improves from 0.09 to 0.35. For the `Dataset3`, the accuracy of the classifier went from 0.18 (no SDG) to 0.7 (with SDG). However, further accuracy improvements were limited by the data distribution being different in the train and validate set because of extremely small number of samples. This would be a good candidate for iterative data quality and classifier performance improvement (Khabiri et al., 2019).

	No SDG	With SDG
Dataset 2 (All)	0.74	0.74
Dataset 2 (7 smallest)	0.09	0.35
Dataset 3 (All)	0.18	0.70

Table 6: Comparison of Macro f_1 average for two industrial datasets. For Dataset 2, we show significant improvement with SDG for smallest classes.

Next we describe the deployment in a commercial environment, and the ensuing challenges and lessons learnt from it.

8 Deployment Solution

Our solution is deployed using the *KModels* infrastructure (Abitbol et al., 2024), which facilitates the encapsulation of both the model generation/prediction code and all relevant metadata within a single model *template*. This metadata delineates the model's input and output requirements, along with various configuration parameters (See

Prompt Type	Include Domain	Include Problem Code	Dataset 2 %Valid Expansions	Dataset 2 Classifier accuracy/ f_1-macro-avg	Dataset 3 %Valid Expansions	Dataset 3 Classifier accuracy/ f_1-macro-avg
No SDG Pipeline	-	-	-	0.57/0.40	-	0.18/0.04
Simple Prompt	-	-	79%	**0.58/0.45**	61%	0.45/0.41
Simple Prompt	y	y	65%	0.57/0.43	66%	0.47/0.41
Detailed Instr.	-	-	83%	0.57/0.43	64%	0.54/0.53
Detailed Instr.	y	y	**87%**	0.57/0.43	**84%**	**0.61/0.60**

Table 5: Pipeline with *only generative LLM* method of SDG, using Google/flan-ul2 models, showing the impact of different prompts. Results show dramatic improvement in the accuracy for Dataset 3 where the number of samples was really small.

example in Figure 4). For example, different sites might want to access their LLM's locally, or from different sites, each site may have different cpu and memory resources, different data connectors, etc. Combined with the model's code, these elements constitute the blueprint for instantiating and deploying a model.

The model training and inferencing code was containerized, and made available in private container registry. The KModels controller is deployed in OpenShift cluster, and is accessible through API. To generate a model, an API request with configuration is sent to controller, which initiates model training, generates a unique model and starts an inferencing server for the generated model. Future inferencing requests are sent to this server.

The model was deployed in a production environment with over 150 sites, and has collected data for over a year. Throughout this duration, the model processed tens of thousands of work orders, yielding positive outcomes (details in Section 8.1). The client's data exhibited distinct geographical patterns, *e.g.* different languages, and the model could be configured to use different translation for the RTT. In addition, different data connections could be facilitated for different sites. The ability to fine-tune the model to local conditions demonstrates the adaptability and scalability of the KModels methodology in real-world industrial applications.

8.1 Deployment Results

And the final validation of the approach came from the actual installation of the tool on the customer site. This client had a particular problem where a large number of work orders were labeled as "miscellaneous", even when they could be assigned a valid class. The model was deployed once the

Figure 4: An example of a template's metadata: On the left, an abstract snippet of the metadata definition for our model's template. On the right, a corresponding configuration instance used in one of the model's deployments.

accuracy of the model was acceptable on the test data. The solution was running for over 6 months with the human override capability. However, we saw that over time, fewer and fewer overrides were made, the class specification of "miscellaneous" gradually decreased from 54% to 17%.

9 Conclusions and Future Work

We presented a novel configurable and extensible automated pipeline for industrial data classification with very little data, and showed an almost 40% improvement in the accuracy for generic and domain specific data. In addition, we provide a KModels framework for nimble deployment of these AI models in a production environment.

Industrial environments continue to present present numerous challenges for AI model development and deployment due to their lack of tolerance for faulty predictions, and hallucinations.

Ethics Statement

We hereby acknowledge that all of the co-authors of this work have gone through the ACL Code of Ethics and honor that code.

References

Roy Abitbol, Eyal Cohen, Muhammad Kanaan, Bhavna Agrawal, Yingjie Lie, Anuradha Bhamidipaty, and Erez Bilgory. 2024. Kmodels: Unlocking ai for business applications. *arXiv preprint*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Elham Khabiri, Wesley M Gifford, Bhanukiran Vinzamuri, Dhaval Patel, and Pietro Mazzoleni. 2019. Industry specific word embedding and its application in log classification. In *Proceedings of the 28th acm international conference on information and knowledge management*, pages 2713–2721.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding. *Advances in Neural Information Processing Systems*, 35:462–477.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. 2021. Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ digital medicine*, 4(1):86.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Yasmen Wahba, Nazim Madhavji, and John Steinbacher. 2023. Attention is not always what you need: Towards efficient classification of domain-specific text: Case-study: It support tickets. In *Science and Information Conference*, pages 1159–1166. Springer.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Dingjun Wu, Jing Zhang, and Xinmei Huang. 2023. Chain of thought prompting elicits knowledge augmentation. *arXiv preprint arXiv:2307.01640*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*.